Non-Covered Code A SW DER's Perspective

July 28, 2005

Rockwell Collins

Topics

- Definitions
- Non-Covered code (aka “unreached” code)
- What to do about non-covered code?
- Summary

Rockwell Collins

Definitions

- Definition: Non-covered code is source code that was not structurally covered during requirement-based testing.
- The non-covered source code can be categorized under three definitions:
  - deactivated,
  - dead and
  - live – Yes, I said “live” – bear with me.
- For Dead and Deactivated code definitions, we need to stick to the DO-178B definitions.

Rockwell Collins

Definitions

- Deactivated code - Executable object code (or data) which “by design” is either
  - (a) not intended to be executed (code) or used (data), for example, a part of a previously developed software component, or
  - (b) is only executed (code) or used (data) in certain configurations of the target computer environment, for example, code that is enabled by a hardware pin selection or software programmed options.

Rockwell Collins

Definitions

- Dead code - Executable object code (or data) which, as a "result of a design error" cannot be executed (code) or used (data) in a operational configuration of the target computer environment and "is not" traceable to a system or software requirement. An exception is embedded identifiers.

Rockwell Collins

Definition: “Live”

- An ‘implicit’ definition for non-covered code that does not fit under the definition of deactivated or dead code.

  Live Code – code that may be executed in the current configuration, at any point in time, depending on the control flow of the program.

- This is the area we seem to be trying to become “inventive” with naming/categorizing - trying to name non-covered code as defensive, default, fail monitor, etc.

Rockwell Collins

What to do about the Non-covered code?

- When Non-Covered code is found, one must ask:

  1st "Why is it there?" (are there requirements for it) and

  2nd "Is it correct?" (with respect to requirements)

- Can't just add a test to cover it (unless part of a requirement that was not tested).

Rockwell Collins

An Aside

- As an aside, we need to keep in mind that SCA is not the goal of verification testing – correct verification of requirements is the goal!
  - SCA is a measurement of how well the code relates to the requirements.
  - SCA identifies code not tested during the high/low level testing.
  - SCA resolution looks to identify code that implements unintended functionality.

Rockwell Collins

What to do about non-covered Deactivated code?

- Two types per DO-178B, **6.4.4.3 Structural Coverage Analysis Resolution**
- d. Deactivated code:
  - For deactivated code which is not intended to be executed in any configuration used within an aircraft or engine, a combination of analysis and testing should show that the means by which such code could be inadvertently executed are prevented, isolated, or eliminated. (Type I)

Rockwell Collins

What to do about non-covered Deactivated code?

- d. Deactivated code: (cont)
  - For deactivated code which is only executed in certain configurations of the target computer environment, the operational configuration needed for normal execution of this code should be established and additional test cases and test procedures developed to satisfy the required coverage objectives. (Type II)

Rockwell Collins

What to do about non-covered Deactivated code?

- Type I – “For deactivated code which is not intended to be executed in any configuration used within an aircraft or engine …”
  - Term “any configuration used within an aircraft or engine” interpreted to mean operational software.
  - This type does not have to be verified; only the deactivation method must be verified.

Rockwell Collins

What to do about non-covered Deactivated code? Type I (cont)

- The DO-178B does not restrict the type of code that can be deactivated and this needs to remain the case.
- Examples might be: bench test code, maintenance code, in-air test data collection, multi-project code, etc.

Rockwell Collins

What to do about non-covered Deactivated code?

- Type II – “For deactivated code which is only executed in certain configurations …”
  - Must be verified as it can become active (operational) without changing any executable code.
  - Activation could happen via modified SW configuration data/HW straps/etc., or activation of functionality by the user via a code entered, etc.

Rockwell Collins

What to do about non-covered Deactivated code?

- For either Type I or II, deactivation needs to be planned - discussed in the PSAC and in plans.
- Both types must have the deactivation method defined in the high or low level requirements and those requirements verified.

Rockwell Collins

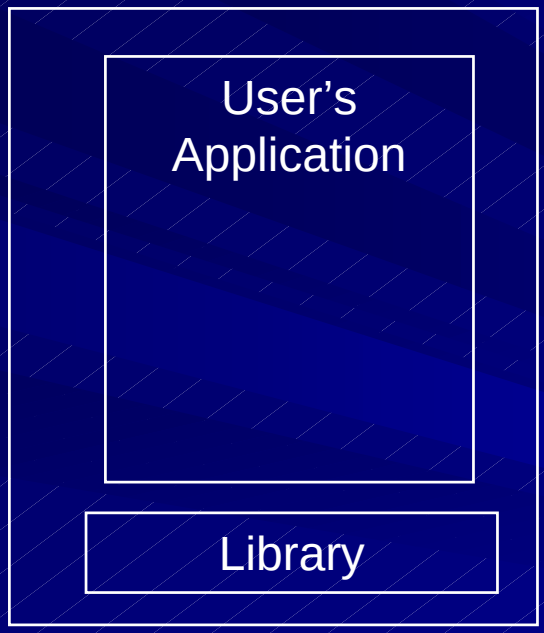
What to do about non-covered Deactivated code?

- There is one area that we noted above that adds a nuance to this – which for discussion purposes we'll call 'procedural' deactivation.
- There are 2 cases for this: 'direct control' and a Reusable SW Component (RSC):
  - Direct control means that the use (activation or deactivation) of functions (e.g., library, etc.) are under the application development team's control.
  - RSC means that the use (activation/deactivation) of a RSC's functions are not under the developer's control.

Rockwell Collins

What to do about non-covered Deactivated code?

Direct Control



Application developer is the same as library developer. Application developer can modify the library.

RSC

Application developer is NOT the same as library developer. Cannot modify the Library.

Rockwell Collins

What to do about non-covered Deactivated code?

- With direct control, the deactivated functions that are left in the software, do not have to be verified, because they are deactivated, since there are to no calls to them.
- A software change must take place in order for the deactivated function to become active and the team will then verify the now active function.
- Caution here in this direct control approach of not verifying, could lead to the possibility of “forgetting” to verify the function later.
- One might want to ask to see how they plan to prevent “forgetting”.

Rockwell Collins

What to do about non-covered Deactivated code?

- With a RSC the development team does not have control of activation/deactivation of the RSC functions.
- Thus, an RSC is different in that functionality can be activated without changing the RSC component.
- The application code developer changes their application code and now there is a call to the RSC function that was deactivated.
- Thus, the deactivated component must be verified unless it is left out of the link making a change to the RSC necessary to activate and access it.

Rockwell Collins

What to do about non-covered Deactivated code?

- In both the direct control and RSC there are no requirements for the deactivation.
- The decision on the handling of the function is an outcome of the development process where the function is not called – thus procedural and not a requirement per se.

Rockwell Collins

What to do about non-covered Dead code?

- DO-178B, **6.4.4.3 Structural Coverage Analysis Resolution**
- c. Dead code: The code should be removed and an analysis performed to assess the effect and the need for re-verification.
- I agree with this, but issues have been encountered with rapid removal of code late in programs.

Rockwell Collins

What to do about non-covered Dead code?

- But problems have been encountered with not removing dead code and the code is executed – it was not dead!
- But problems have been encountered with removing code and the code was needed – it was not dead!
- So, now what?

Rockwell Collins

What to do about non-covered Dead code?

- The analysis performed in making the dead code determination in each case was flawed.
- The development team and Certification review should focus on:
  - "How thorough was the analysis that made the determination?"
  - "Was the analysis reviewed by peers?"
  - "Is it understandable and do we agree with it?"

Rockwell Collins

What to do about non-covered Dead code?

- If the analysis is planned, complete and assured correct, then the software should be safe and the removal is a 2nd order effort.
- The dead code then can be identified in a problem report and a schedule for removal negotiated with the certification authorities.
- As a DER, you need review to see that the team has addressed dead code as part of their formal verification process.

Rockwell Collins

What to do about non-covered Live code?

- Live – this would seem to be straightforward for non-covered code. DO-178B Structural Coverage Analysis states:
- **6.4.4.3 Structural Coverage Analysis Resolution**
  - Structural coverage analysis may reveal code structure that was not exercised during testing. Resolution would require additional software verification process activity. This unexecuted code structure may be the result of:

Rockwell Collins

What to do about non-covered Live code?

- a. Shortcomings in requirements-based test cases or procedures: The test cases should be "supplemented" or test procedures changed to provide the missing coverage. The method(s) used to perform the requirements-based coverage analysis may need to be reviewed.
- b. Inadequacies in software requirements: The software requirements should be modified and additional test cases developed and test procedures executed.

Rockwell Collins

What to do about non-covered Live code?

- Let's add an implicit additional analysis resolution statement:
- **6.4.4.3 Structural Coverage Analysis Resolution**
  - "e." If the code does not belong in the system (i.e., there is no requirement for this code to be in system – unintended functionality), it must be removed. (Note, this is not dead code, because the code is live and but not executed by a requirements-based test.)

Rockwell Collins

What to do about non-covered Live code?

- There is the question of test vs. analysis of non-covered “live” code.
  - Both are acceptable verification methods.
  - Analysis is used to assure complex issues such as partitioning and other safety-related requirements, so should be acceptable for non-covered code assurance

Rockwell Collins

What to do about non-covered Live code?

- DERs need to review the analyses of the non-covered code to see if they are thorough, complete and assure the analysis addresses that the code is implementing a requirement
- i.e., DERs need to assure the team has stated:
  - "The code is there due to requirement X."
  - "If executed, the code would work correctly per the requirement."

Rockwell Collins

What to do about non-covered Live code?

- Note that for some constructs such as a default at the end of a case statement, I allow the requirement to come from the Design or Coding standard.
- If the team does use the standard, I asked further that they define what the behavior should be. Examples:
  - If the requirement is to pick out some specific labels from an incoming 429 stream of data, then the default case simply can drop the others.
  - If the requirement is to switch on modes and there are 4, then what if a 5th is encountered. Can just discard as above.

Rockwell Collins

What to do about non-covered Live code?

- Bottom line - if the code is active, then there must be a related requirement and the code verified.
- If there is no requirement for the code to be there, it should be removed (like dead code).
  - In this case, I feel similar rules need to apply as in the case of dead code, so code is not removed late in the program right before delivery.
  - Again need to emphasize analysis to assure the code is really “unintended functionality” and then a planned, reviewed and negotiated removal can occur.

Rockwell Collins

Summary

- Definitions should follow DO-178B.
- Categorization may help in aligning all parties.
- Any categorization should use the three main definitions.
- Guidance is needed from the certification authorities more for analysis assurance concerns than for definitions.

Rockwell Collins

Summary

- Discussion/Questions

Rockwell Collins